UNIVERSIDAD DE GUAYAQUIL

FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

AUMENTO DE LA DISPONIBILIDAD DE BASES DE DATOS MYSQL UTILIZANDO UNA CONFIGURACIÓN DE BASE EN MODO DE ESPERA E INSTALACIÓN DE AGENTES PARA AUTOMATIZAR CONEXIONES

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

AUTOR: POLL MIGUEL CEDEÑO TELLO

TUTOR: ING. RAYNAUD QUIROZ

GUAYAQUIL – ECUADOR

2011

Guayaquil, 11 de Octubre del 2010

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del trabajo de investigación, “Aumento de la disponibilidad de bases de datos MySQL utilizando una configuración de base en modo de espera e instalación de agentes para automatizar conexiones “elaborado por el Sr. Poll Miguel Cedeño Tello, egresado de la Carrera de Ingeniería en Sistemas Computacionales, Facultad de Ciencias Matemáticas y Físicas de la Universidad de Guayaquil, previo a la obtención del Título de Ingeniero en Sistemas, me permito declarar que luego de haber orientado, estudiado y revisado, la Apruebo en todas sus partes.

Atentamente

……

Ing. Raynaud Quiroz

TUTOR

DEDICATORIA

Agradezco a Dios que me ha dado la fuerza y capacidad para llegar hasta aquí, así como a mi familia que me ha estado apoyando en toda la trayectoria de mi carrera tanto en las buenas como en las malas.

TRIBUNAL DE GRADO

Ing. Fernando Abad Montero DECANO DE LA FACULTAD CIENCIAS MATEMÁTICAS Y FÍSICAS	Ing. Juan Chanabá Alcócer DIRECTOR
Ing. Raynaud Quiroz TUTOR	Inga. Ana María Arellano PROFESOR DEL ÁREA - TRIBUNAL
Ing. Juan Carlos Cedeño PROFESOR DEL ÁREA – TRIBUNAL	Dr. José Júpiter SECRETARIO

UNIVERSIDAD DE GUAYAQUIL

FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

AUMENTO DE LA DISPONIBILIDAD DE BASES DE DATOS MYSQL UTILIZANDO UNA CONFIGURACIÓN DE BASE EN MODO DE ESPERA E INSTALACIÓN DE AGENTES PARA AUTOMATIZAR CONEXIONES

MySQL es una de las bases de datos libres y de código abierto más usadas en muchos ambientes laborales. Tiene un sin número de herramientas útiles y para cada lugar de trabajo. Este proyecto busca mejorar una de las áreas de MySQL que es la replicación. El problema se debe a que el método de replicación es asíncrono esto significa que no es inmediato y que se debe esperar cierta cantidad de tiempo para que la información se replique, y este tiempo va a depender de varios factores como pueden ser hardware, software, configuraciones; por estos motivos se puede dar el caso de que al momento de replicar la información no llegue o llegue incompleta. La solución que se plantea consiste en extraer del cliente las consultas que se envían a la base de datos y guardarlos en archivos; para lograr esto se procedió con la edición del código fuente del ODBC, agregando también una función la cual va a estar indicando si las consultas se están ejecutando correctamente; caso contrario si alguna no se puede ejecutar se escribirá un mensaje de error en un registro que llevará el nombre del DSN. Se procederá a la instalación de agentes tanto en los clientes como en un servidor esclavo. El agente que se encontrará en el servidor esclavo estará censando la comunicación con el maestro, si se llegara a perder la comunicación se realizarán las configuraciones necesarias para que el servidor funcione como maestro para así poder recibir las consultas enviadas por todos los clientes. El agente en el cliente va a estar encargado de leer el registro con el nombre del DSN; al detectar algún mensaje de error procederá a realizar procesos para replicar la información almacenada, para que así no exista pérdida de esta.

UNIVERSIDAD DE GUAYAQUIL

FACULTAD DE CIENCIAS MATEMÁTICAS Y FÍSICAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

AUMENTO DE LA DISPONIBILIDAD DE BASES DE DATOS MYSQL UTILIZANDO UNA CONFIGURACIÓN DE BASE EN MODO DE ESPERA E INSTALACIÓN DE AGENTES PARA AUTOMATIZAR CONEXIONES

MySQL is one of the free data base and open source used in many workplaces. It has different useful tools for each Workplace. This project seeks to improve one of the area of MySQL that is the replication. The problem is that the method of replication is asynchronous this means that is not immediate and it needs to expect a certain amount of time in order to that this information can be replicated and this time will depend on several factors such as hardware, software, configurations, for these reasons it may be the case when comes the moment to replicate the information, this information does not arrive or arrive incomplete. The solution that arises is to extract customer queries that are send to the database and save in files; to achieve this we proceeded with the release of the ODBC code source, also adding a function which will be indicate if queries are running correctly, otherwise if you cannot run it will write an error message in a register held by the name of the DSN. It will install both agents clients like a slave server. The agent may be found on the slave communication registers you with the teacher, if you were to lose the communication was made necessary settings for the server to work as a teacher in order to receive the queries sent by all clients. The agent on the client will be responsible for reading the record with the name of the DSN, to detect any error messages shall carry out processes to replicate the information stored, so that no loss of it.

CERTIFICADO DE ACEPTACIÓN DEL TUTOR

En mi calidad de Tutor del Primer Curso de Fin de Carrera, nombrado por el Departamento de Graduación y la Dirección de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil,

CERTIFICO:

Que he analizado el Proyecto de Grado presentado por el/la egresado(a) Poll Miguel Cedeño Tello, como requisito previo para optar por el título de Ingeniero cuyo problema es:

AUMENTO DE LA DISPONIBILIDAD DE BASES DE DATOS MYSQL UTILIZANDO UNA CONFIGURACIÓN DE BASE EN MODO DE ESPERA E INSTALACIÓN DE AGENTES PARA AUTOMATIZAR CONEXIONES

considero aprobado el trabajo en su totalidad.

Presentado por:

Poll Miguel Cedeño Tello	0922634670
Apellidos y Nombres completos	Cédula de ciudadanía N°

Tutor: Ing. Raynaud Quiroz

Guayaquil, _____de_____

ÍNDICE GENERAL

CARÁTULA	i
CARTA DE ACEPTACIÓN DEL TUTOR	ii
DEDICATORIA	iii
TRIBUNAL	iv
RESUMEN	v
ABSTRACT	vi
CERTIFICADO DE APROBACIÓN	vii
ÍNDICE GENERAL	viii
ÍNDICE DE GRÁFICOS	ix
RESOLUCIÓN DEL PROYECTO	1
MANUAL DE USUARIO DEL ODBC	12
MANUAL TÉCNICO DEL ODBC	15
MANUAL DE USUARIO DEL AGENTE	30
MANUAL TÉCNICO DEL AGENTE	33

ÍNDICE DE GRÁFICOS

Pág.

Gráfico 1 3
Estado del Maestro “Show Master Status”

Gráfico 2 5
Base de datos ‘data’ creada

Gráfico 3 6
Base de datos ‘data’ replicada

Gráfico 4 7
Plug-in’s instalados

Gráfico 5 9
Replicación semi-síncrona maestro.

Gráfico 6 9
Replicación semi-síncrona esclavo

Gráfico 7 10
Registro de la base de datos Maestro

Gráfico 8 12
Configurando el conector ODBC

Gráfico 9 13
Activando la opción “Log queries by PID and DSN”

Gráfico 10 14
Archivos donde se guardan las transacciones y el log del DSN.

Gráfico 11 16
Abriendo el código fuente del ODBC

Gráfico 12 16
Sección DEBUG en ODBC

Gráfico 13 18
Checkbox para las sentencias creado

Gráfico 14 24
Registro agregado en Windows

Gráfico 15 29
Archivos Creados

RESOLUCIÓN DEL PROYECTO

INICIANDO REPLICACIÓN EN MYSQL

Se debe tomar en cuenta que tanto el maestro como el esclavo se puedan comunicar por el puerto 3306, que es el puerto por defecto con el que trabaja MySQL. También se asumirá que ha sacado respaldo del maestro y los ha ejecutado en el esclavo en el caso de que tenga información ingresada.

También que el servidor usado como de replicación exista en el archivo /etc/hosts.

La replicación presente es la llamada maestro – esclavo que es la básica, y hay que recordar que la replicación es asíncrona. El tiempo de replicación por defecto es de 60 segundos.

La IP asignada al maestro va a ser 192.168.100.10 y la del esclavo 192.168.100.20

Los siguientes pasos serán para el servidor maestro.

En el archivo /etc/my.cnf debemos tener los siguientes valores:

```
log-bin = proyecto-bin

server-id=1
```

```
bin-address=0.0.0.0 #Para permitir conexiones remotas

skip-name-resolve #Si no se usa dominio
```

Ahora procedemos a reiniciar el servidor MySQL para que lea las opciones dadas.

Ingresemos a MySQL y creemos una cuenta para que el servidor de replicación se pueda conectar a la base de datos.

```
mysql> grant replication slave on *.* to ‘mysqlrepl’@’%.proyecto.com’

    -> identified by ‘replica86’;
```

Con esto hemos creado un usuario de nombre “mysqlrepl” con contraseña “replica86” el cual solo va a servir para la replicación.

A continuación ejecutamos los siguientes comandos:

```
mysql> flush tables with read lock;

mysql> show master status;
```

En este momento vamos a tener la siguiente salida.

Gráfico 1. Estado del Maestro “Show Master Status”

```
mysql> show master status
    -> ;
+----------------------+----------+--------------+------------------+
| File                 | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+----------------------+----------+--------------+------------------+
| proyecto-bin.000013 |      509 |              |                  |
+----------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

Aquí copiamos los valores de File y Position

Ejecutamos el comando:

```
shell> unlock tables;
```

Para continuar con la escritura de datos en el maestro. Estos son los pasos necesarios ahora procedemos con el esclavo.

En el archivo /etc/my.cnf del esclavo debemos tener lo siguiente.

```
[mysqld]
```

```
server-id=2
```

Reiniciamos el servicio de mysql.

Ahora desde el cliente de mysql del cliente ejecutamos el siguiente comando.

```
mysql> change master to
```

```
->MASTER_HOST=’proyecto.com’,

->MASTER_USER=’mysqlrepl’,

->MASTER_PASSWORD=’replica86’,

->MASTER_LOG_FILE=’proyecto_bin.000013’

->MASTER_LOG_POS=509;
```

Con esto indicamos los parámetros de conexión al maestro.

Por último iniciamos el esclavo con:

```
mysql> start slave;
```

Para probar que todo esté funcionando crearemos una base de datos e importaremos un respaldo. En mysql procedemos a crear una base de datos

```
mysql> create database data;
```

Ahora desde una terminal de Linux importamos el archivo de creación. Para este ejemplo nos vamos a ubicar dentro del directorio de instalación de mysql (/usr/local/mysql)

```
# bin/mysql –u root –p data < /super.sql
```

Donde /super.sql es donde se encuentra el respaldo. Si todo ha funcionado correctamente no nos debe presentar ningún error. Ahora ingresamos a MySQL para comprobar que se importaron correctamente los datos.

```
mysql> use data;
```

```
mysql> show tables;
```

Con estos comandos procedemos a ver la lista de tablas creadas dentro de la base de datos data. Como la replicación ya ha sido iniciada encontraremos la base de datos

Gráfico 2. Base de datos 'data' creada

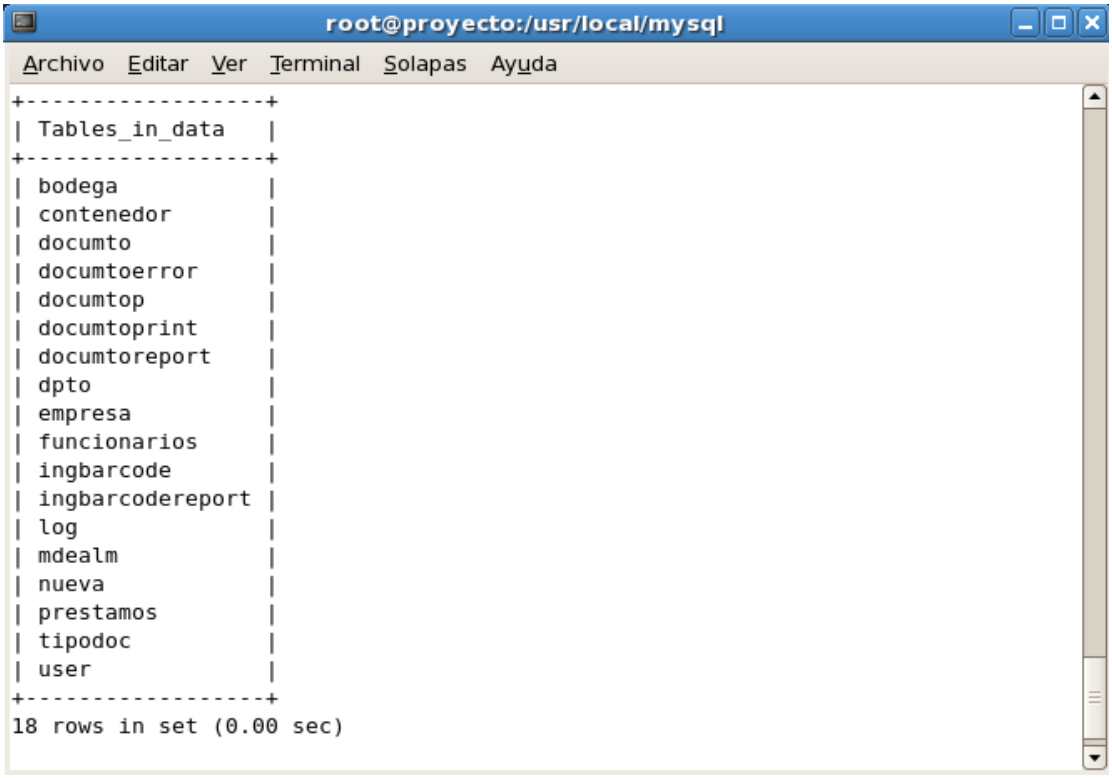
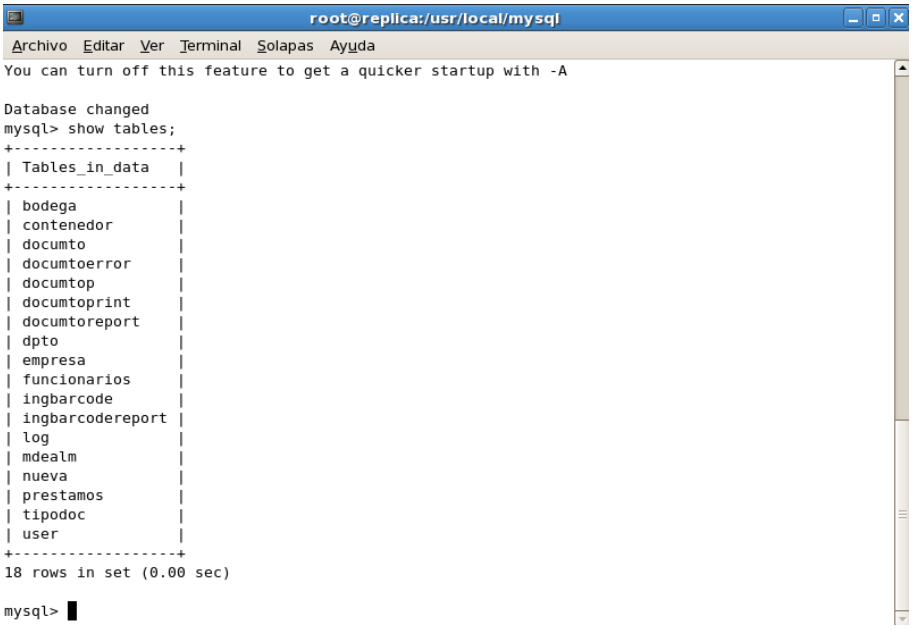


Gráfico 3. Base de datos ‘data’ replicada



REPLICACIÓN SEMI-SÍCRONA

Ahora vamos a iniciar con la replicación semi-síncrona a diferencia de la replicación normal esta tiene un tiempo para replicar por defecto de 10 segundos.

Lo primero que se debe hacer es instalar el Plug-in.

En el maestro:

```
mysql> install plugin rpl_semi_sync_master soname ‘semisync_master.so’;
```

Si este comando nos devuelve algún tipo de error debemos revisar que el Plug-in se encuentre en el siguiente directorio /usr/local/mysql/lib/plugin.

Ahora en el esclavo también lo instalamos pero el que corresponde al esclavo.

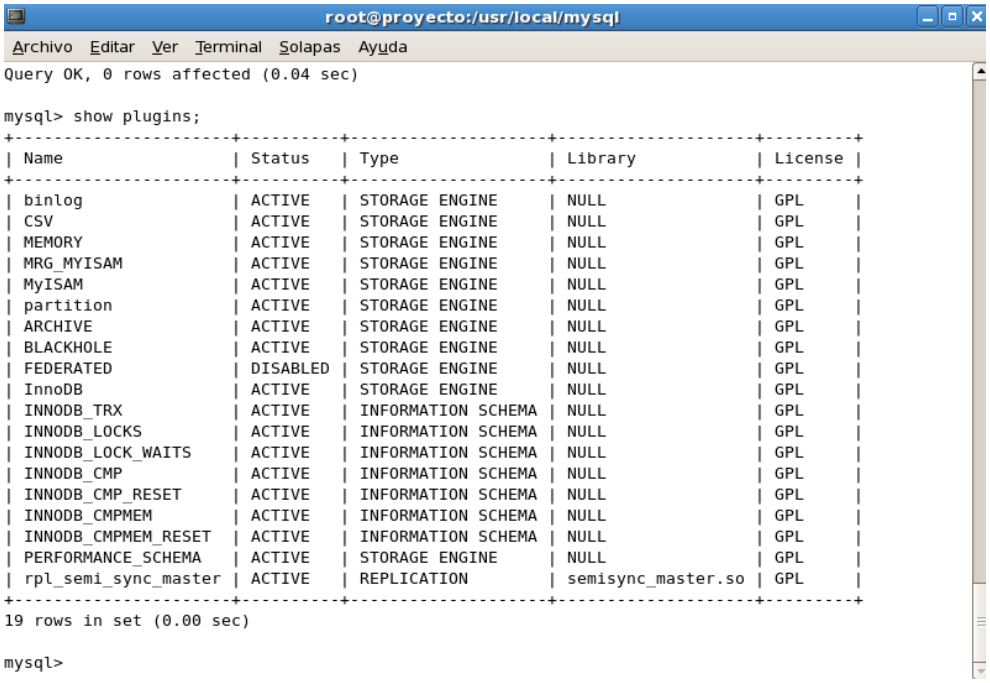
```
mysql> install plugin rpl_semi_sync_slave soname ‘semisync_slave.so’;
```

Para ver los Plug-in’s instalados ejecutamos el siguiente comando;

```
mysql> show plugins;
```

En el listado que se muestra a continuación podemos ver instalado el Plug-in resiente.

Gráfico 4. Plug-in’s instalados


root@proyecto:/usr/local/mysql
Archivo Editar Ver Terminal Solapas Ayuda
Query OK, 0 rows affected (0.04 sec)
mysql> show plugins;
Name	Status	Type	Library	License
binlog	ACTIVE	STORAGE ENGINE	NULL	GPL
CSV	ACTIVE	STORAGE ENGINE	NULL	GPL
MEMORY	ACTIVE	STORAGE ENGINE	NULL	GPL
MRG_MYISAM	ACTIVE	STORAGE ENGINE	NULL	GPL
MyISAM	ACTIVE	STORAGE ENGINE	NULL	GPL
partition	ACTIVE	STORAGE ENGINE	NULL	GPL
ARCHIVE	ACTIVE	STORAGE ENGINE	NULL	GPL
BLACKHOLE	ACTIVE	STORAGE ENGINE	NULL	GPL
FEDERATED	DISABLED	STORAGE ENGINE	NULL	GPL
InnoDB	ACTIVE	STORAGE ENGINE	NULL	GPL
INNODB_TRX	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_LOCKS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_LOCK_WAITS	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_CMP	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_CMP_RESET	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_CMPMEM	ACTIVE	INFORMATION SCHEMA	NULL	GPL
INNODB_CMPMEM_RESET	ACTIVE	INFORMATION SCHEMA	NULL	GPL
PERFORMANCE_SCHEMA	ACTIVE	STORAGE ENGINE	NULL	GPL
rpl_semi_sync_master	ACTIVE	REPLICATION	semisync_master.so	GPL
19 rows in set (0.00 sec)
mysql>

Ahora se debe habilitar las variables globales de esta replicación ya que por defecto están deshabilitadas.

En el maestro se ejecuta lo siguiente;

```
mysql> set global rpl_semi_sync_master_enabled = 1;

mysql> set global rpl_semi_sync_master_timeout = 1000;
```

Con esto le decimos que esta activada la replicación semi-síncrona y que el tiempo va a ser de 1 segundo.

Ahora en el esclavo ejecutamos:

```
mysql> set global rpl_semi_sync_slave_enabled = 1;
```

Al igual que en el maestro se debe indicar que la replicación semi-síncrona está activada.

Ahora debemos re-inicializar el hilo de entrada y salida del esclavo.

```
mysql> stop slave io_thread; start slave io_thread;
```

Ahora procederemos a hacer una prueba para comprobar el funcionamiento como estamos en un ambiente de prueba la replicación debe hacerse casi inmediata, por esto vamos a hacer 3 sentencias en el maestro.;

Gráfico 5. Replicación semi-síncrona maestro

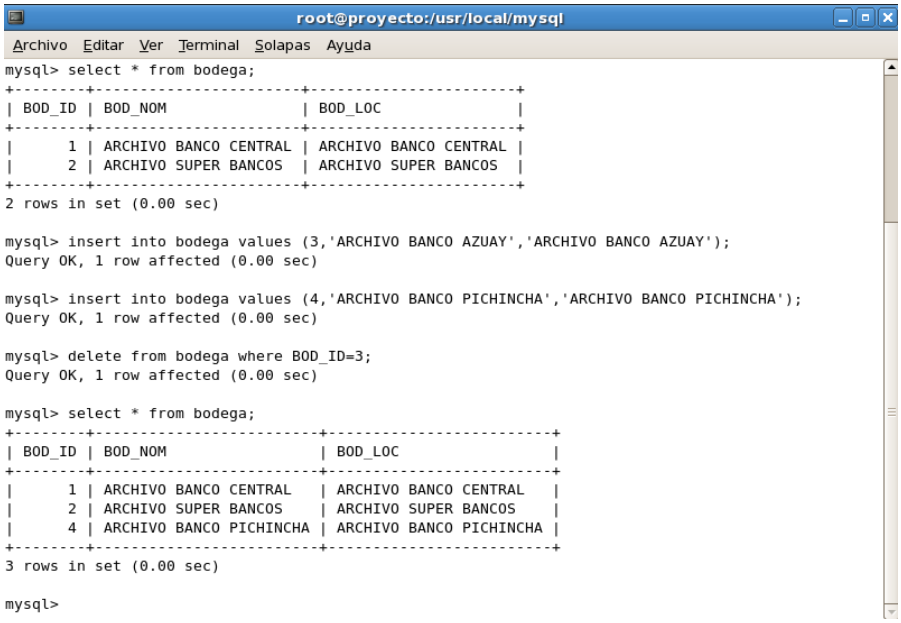
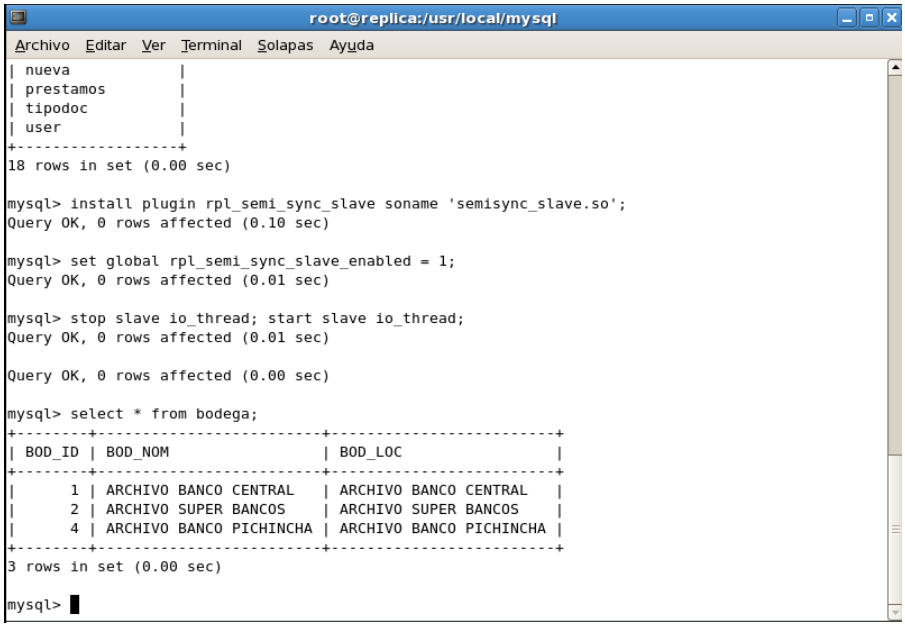
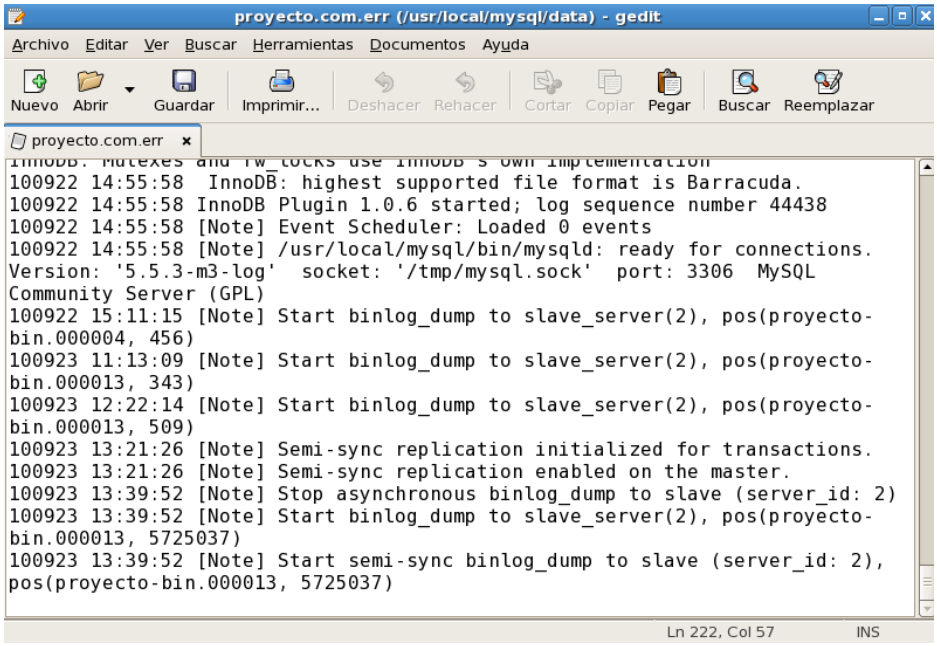

root@proyecto:/usr/local/mysql
Archivo Editar Ver Terminal Solapas Ayuda
mysql> select * from bodega;
+--------+-----------------------+-----------------------+
| BOD_ID | BOD_NOM | BOD_LOC |
+--------+-----------------------+-----------------------+
| 1 | ARCHIVO BANCO CENTRAL | ARCHIVO BANCO CENTRAL |
| 2 | ARCHIVO SUPER BANCOS | ARCHIVO SUPER BANCOS |
+--------+-----------------------+-----------------------+
2 rows in set (0.00 sec)
mysql> insert into bodega values (3,'ARCHIVO BANCO AZUAY','ARCHIVO BANCO AZUAY');
Query OK, 1 row affected (0.00 sec)
mysql> insert into bodega values (4,'ARCHIVO BANCO PICHINCHA','ARCHIVO BANCO PICHINCHA');
Query OK, 1 row affected (0.00 sec)
mysql> delete from bodega where BOD_ID=3;
Query OK, 1 row affected (0.00 sec)
mysql> select * from bodega;
+--------+-------------------------+-------------------------+
| BOD_ID | BOD_NOM | BOD_LOC |
+--------+-------------------------+-------------------------+
1	ARCHIVO BANCO CENTRAL	ARCHIVO BANCO CENTRAL
2	ARCHIVO SUPER BANCOS	ARCHIVO SUPER BANCOS
4	ARCHIVO BANCO PICHINCHA	ARCHIVO BANCO PICHINCHA
+--------+-------------------------+-------------------------+
3 rows in set (0.00 sec)
mysql>

Gráfico 6. Replicación semi-síncrona esclavo


root@replica:/usr/local/mysql
Archivo Editar Ver Terminal Solapas Ayuda
| nueva |
| prestamos |
| tipodoc |
| user |
+-------------------+
18 rows in set (0.00 sec)
mysql> install plugin rpl_semi_sync_slave soname 'semisync_slave.so';
Query OK, 0 rows affected (0.10 sec)
mysql> set global rpl_semi_sync_slave_enabled = 1;
Query OK, 0 rows affected (0.01 sec)
mysql> stop slave io_thread; start slave io_thread;
Query OK, 0 rows affected (0.01 sec)
Query OK, 0 rows affected (0.00 sec)
mysql> select * from bodega;
+--------+-------------------------+-------------------------+
| BOD_ID | BOD_NOM | BOD_LOC |
+--------+-------------------------+-------------------------+
1	ARCHIVO BANCO CENTRAL	ARCHIVO BANCO CENTRAL
2	ARCHIVO SUPER BANCOS	ARCHIVO SUPER BANCOS
4	ARCHIVO BANCO PICHINCHA	ARCHIVO BANCO PICHINCHA
+--------+-------------------------+-------------------------+
3 rows in set (0.00 sec)
mysql>

Claro que no es sencillo apreciarlo mediante imágenes, así que procedemos a mostrar a mostrar el archivo donde se guardan los cambios, este se puede encontrar de la carpeta /usr/local/mysql/data en este caso el nombre del archivo es proyecto.com.err donde proyecto es el nombre del maestro.

Gráfico 7. Registro de la base de datos Maestro



En las dos últimas líneas podemos apreciar que la replicación semi-síncrona fue iniciada con el servidor esclavo.

Si en algún momento queremos reiniciar el servidor y no perder las configuraciones realizadas debemos agregar en el archivo /etc/my.cnf

En el maestro

```
[mysqld]

rpl_semi_sync_master_enabled = 1

rpl_semi_sync_master_timeout = 1000
```

En el esclavo

```
[mysqld]

rpl_semi_sync_slave_enabled = 1
```

MANUAL DE USUARIO DEL ODBC

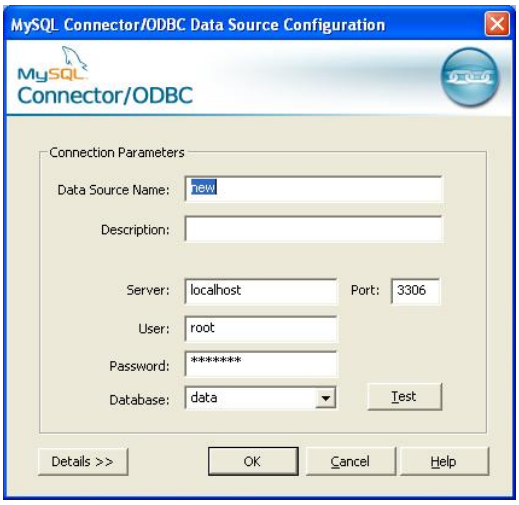
En esta sección se va a proceder a explicar el correcto uso del ODBC modificado en este proyecto. Sólo nos vamos a concentrar en lo que se le ha agregado.

Entre los archivos que tenemos del conector ODBC de MySQL vamos a encontrar un archivo llamado “install.bat” ejecutamos este archivo el cual va a proceder a la instalación del driver de conexión.

La instalación creará un directorio llamado log_queries en el disco c:\ en donde se guardarán los registros de todas las transacciones realizadas por el cliente al servidor.

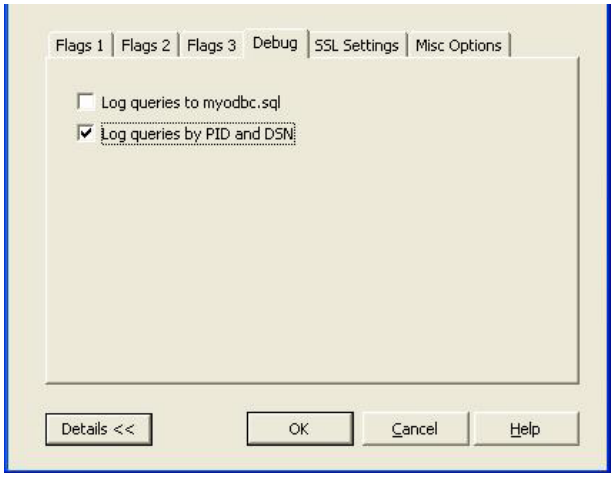
Ahora procedemos a crear un DSN normalmente y damos clic en el botón Details:

Gráfico 8. Configurando el conector ODBC



Se nos desplegará un anexo a la ventana conteniendo varias pestañas. Seleccionamos la pestaña “Debug” y damos click para activar la casilla “Log queries by PID and DSN”, finalmente damos click en el botón “OK” para guardar las configuraciones realizadas.

Gráfico 9. Activando la opción “Log queries by PID and DSN”

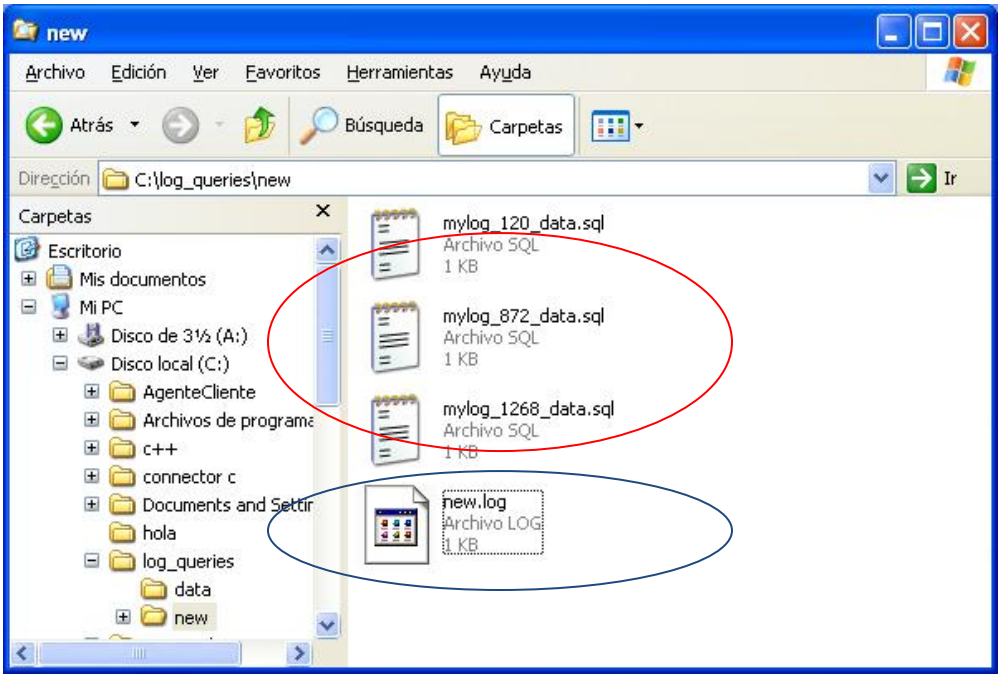


Dentro del directorio c:\log_queries crearemos un nuevo directorio (nueva carpeta) con el mismo nombre del DSN que creamos anteriormente; aquí se guardarán todas las transacciones generadas en archivos con extensión .sql, y sus nombres van a estar dados de la siguiente manera: “mylog**_NombreDeLaBaseDeDatos_PIDdelPrograma**”.

El PID de un programa es el identificador del proceso con que se ejecuta el programa.

Dentro de la carpeta también se creará un archivo con el nombre del DSN y de extensión .log. Este archivo nos indicará si las transacciones están siendo ejecutadas correctamente grabando el estado de “TRANSACCION OK”, y en caso de que haya habido un error se grabará el estado “ERROR”.

Gráfico 10. Archivos donde se guardan las transacciones y el log del DSN.



MANUAL TÉCNICO DEL ODBC

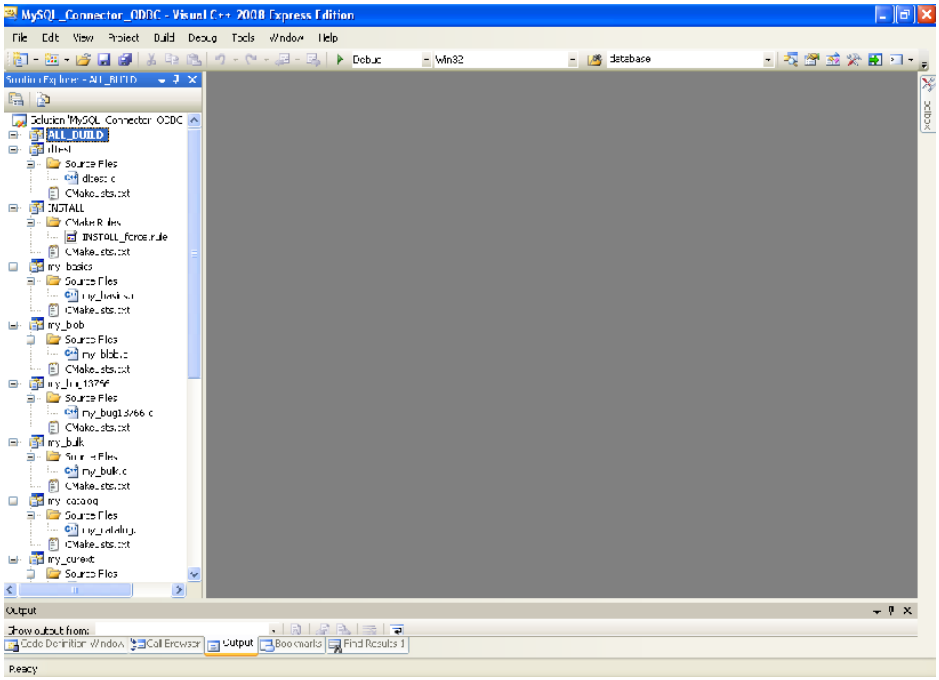
Lo que se plantea hacer como parte de la solución es usar el ODBC para extraer la información que envían los usuarios hacia el servidor. Sólo se explicará lo que se hizo para este proyecto ya que el conector ODBC es un proyecto grande que fue iniciado desde hace varios años.

El compilador que se va a usar es el de Visual Studio 2008 Express Edition.

Luego de haber usado el CMAKE para hacer las configuraciones necesarias procedemos a abrir el archivo.

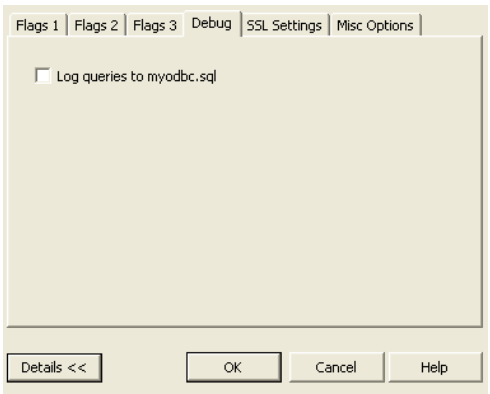
Abrimos el archivo “MySQL_Connector_ODBC.sln” aquí veremos del lado izquierdo dentro de las raíces del árbol las librerías y los archivos en C necesarios para compilar el conector.

Gráfico 11. Abriendo el código fuente del ODBC



Ahora lo primero vamos a poner la opción que vamos a crear dentro la sección Debug en Details del ODBC ya que se relaciona con lo que vamos a hacer.

Gráfico 12. Sección DEBUG en ODBC



Primero en el archivo “C:\mysql-connector-odbc-5.1.6\setupgui\windows\resource.h” agregamos una variable para el checkbox siguiendo el formato de variables creadas.

```
#define IDC_CHECK_save_queries_by_pid_db 10040
```

Ya que los archivos que se van a guardar deben diferenciarse por la aplicación que se usa y por la base de datos, para que no interfiera en las sentencias generada por otras aplicaciones.

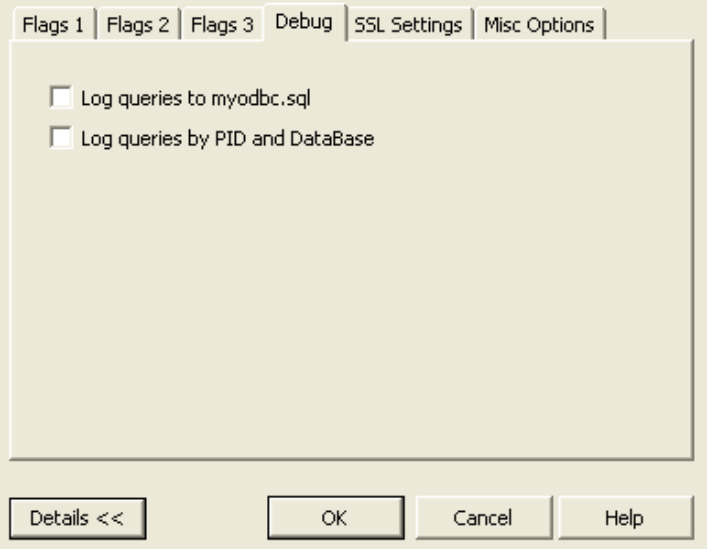
Ahora procedemos a crear el checkbox dentro de la pestaña Debug, para esto editamos el archivo “C:\mysql-connector-odbc-5.1.6\setupgui\windows\ odbcdialogparams.rc”

```
IDD_TAB4 DIALOGEX 0, 0, 209, 151

STYLE DS_SETFONT | DS_FIXEDSYS | WS_CHILD

FONT 8, "MS Shell Dlg", 400, 0, 0x1

BEGIN

    CONTROL         "Log queries to myodbc.sql",IDC_CHECK_save_queries,

                    "Button",BS_AUTOCHECKBOX | WS_TABSTOP,12,12,148,10
```

```
CONTROL "Log queries by PID and
DataBase",IDC_CHECK_save_queries_by_pid_db,
"Button",BS_AUTOCHECKBOX | WS_TABSTOP,12,26,148,10
 END
```

Con esto tenemos listo el checkbox si queremos compilamos para visualizarlo, hay que tomar en cuenta que en estos momentos no hace nada.

Gráfico 13. Checkbox para las sentencias creado.



Dentro del archivo “C:\mysql-connector-odbc-5.1.6\util\ installer.h” en la estructura DataSource creamos la variable:

```
BOOL save_queries_by_pid_db;
```

Esta nos servirá para obtener y escribir el valor en los registros.

En el mismo archivo creamos la macro:

```
#define FLAG_LOG_QUERIES_BY_PID_DB (FLAG_SAFE << 7)
```

Esta se usará para el control del checkbox.

Ahora hay que proceder a grabar en los registros del sistema si se activa esta función, para esto primero creamos el nombre del registro editando el archivo "C:\mysql-connector-odbc-5.1.6\util\ installer.c"

```
static SQLWCHAR W_LOG_QUERY[]=
  {'L','O','G','_','Q','U','E','R','Y',0};
static SQLWCHAR W_LOG_QUERIES_BY_PID_DB[]=
{'L','O','G','_','Q','U','E','R','I','E','S','_','B','Y','_','P','I','D','_','D','B',0};
static SQLWCHAR W_NO_CACHE[]=
  {'N','O','_','C','A','C','H','E',0};
```

Con el nombre que se le asigna a la variable es el con el cual se va a guardar en los registros.

Ahora en el mismo archivo procedemos a agregar a la cadena que escribe en los registros nuestra variable creada.

```
static const

SQLWCHAR *dsnparams[]= {W_DSN, W_DRIVER, W_DESCRIPTION, W_SERVER,

                        W_UID, W_PWD, W_DATABASE, W_SOCKET, W_INITSTMT,

                        W_PORT,          W_OPTION,          W_CHARSET,
                        W_REPLICATION_SERVER, W_SSLKEY,

                        W_SSLCERT, W_SSLCA, W_SSLCAPATH, W_SSLCIPHER,

                        W_SSLVERIFY, W_READTIMEOUT, W_WRITETIMEOUT,

                        W_FOUND_ROWS, W_BIG_PACKETS, W_NO_PROMPT,

                        W_DYNAMIC_CURSOR,                  W_NO_SCHEMA,
                        W_NO_DEFAULT_CURSOR,  W_NO_LOCALE,  W_PAD_SPACE,
                        W_FULL_COLUMN_NAMES,         W_COMPRESSED_PROTO,
                        W_IGNORE_SPACE,W_NAMED_PIPE,W_NO_BIGINT,
                        W_NO_CATALOG,                      W_USE_MYCNF,
                        W_SAFE,W_NO_TRANSACTIONS,          W_LOG_QUERY,
                        W_LOG_QUERIES_BY_PID_DB, W_NO_CACHE,

                        W_FORWARD_CURSOR,             W_AUTO_RECONNECT,
                        W_AUTO_IS_NULL,W_ZERO_DATE_TO_MIN,
                        W_MIN_DATE_TO_ZERO,
```

```
                        W_MULTI_STATEMENTS, W_COLUMN_SIZE_S32,
                        W_NO_BINARY_RESULT,     W_DFLT_BIGINT_BIND_STR,
                        W_ACTIVE_SERVER};
```

En el mismo archivo dentro de la función

```
void ds_map_param(DataSource *ds, const SQLWCHAR *param, SQLWCHAR
***strdest, unsigned int **intdest, BOOL **booldest)
```

Agregamos lo siguiente:

```
else if (!sqlwcharcasecmp(W_LOG_QUERY, param))
    *booldest= &ds->save_queries;
  else if (!sqlwcharcasecmp(W_LOG_QUERIES_BY_PID_DB, param))
      *booldest= &ds->save_queries_by_pid_db;
  else if (!sqlwcharcasecmp(W_NO_CACHE, param))
    *booldest= &ds->dont_cache_result;
```

En la función

```
int ds_add(DataSource *ds)
```

Agregamos:

```
if (ds_add_intprop(ds->name, W_LOG_QUERY, ds->save_queries)) goto error;
```

```
if (ds_add_intprop(ds->name, W_LOG_QUERIES_BY_PID_DB, ds->save_queries_by_pid_db)) goto error;
```

```
if (ds_add_intprop(ds->name, W_NO_CACHE, ds->dont_cache_result)) goto error;
```

En la función

```
void syncTabs(HWND hwnd, DataSource &params)
```

Agregamos

```
ds->save_queries=           (options & FLAG_LOG_QUERY) > 0;
```

```
ds->save_queries_by_pid_db= (options & FLAG_LOG_QUERIES_BY_PID_DB) > 0;
```

```
ds->dont_cache_result=      (options & FLAG_NO_CACHE) > 0;
```

En la función

```
ulong ds_get_options(DataSource *ds)
```

Agregamos

```
  if (ds->save_queries)
```

```
    options|= FLAG_LOG_QUERY;
  if (ds->save_queries_by_pid_db)
    options|= FLAG_LOG_QUERIES_BY_PID_DB;
  if (ds->dont_cache_result)
    options|= FLAG_NO_CACHE;
```

Ahora en el archivo

“C:\mysql-connector-odbc-5.1.6\setupgui\windows\odbcdialogparams.cpp”

En la función:

```
void syncTabsData(HWND hwnd, DataSource &params)
```

Agregamos:

```
/* debug*/
GET_BOOL(4,save_queries);
GET_BOOL(4,save_queries_by_pid_db);
```

Esto nos permite obtener el valor de esta variable, el número 4 es por el valor que tiene la pestaña Debug

En la función

```
void syncTabs(HWND hwnd, DataSource &params)
```

Agregamos

```
SET_BOOL(DEBUG_TAB,save_queries_by_pid_db);
```

En el archivo

C:\mysql-connector-odbc-5.1.6\installer\myodbc3i.c

En la función

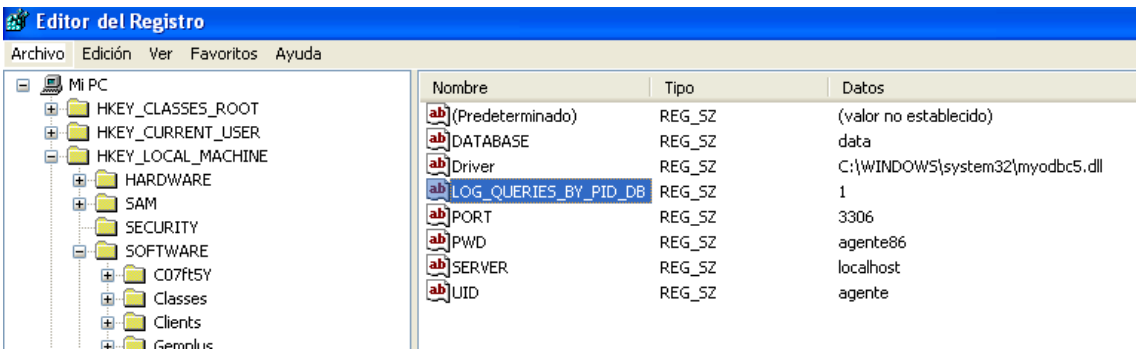
```
int list_datasource_details(DataSource *ds)
```

Agregamos:

```
if (ds->save_queries_by_pid_db) printf("\tLOG_QUERIES_BY_PID_DB\n");
```

Estos son los cambios necesarios para guardar y extraer de los registros el valor del checkbox al momento de activar esta opción.

Gráfico 14. Registro agregado en Windows



Ahora hay que escribir lo que van a hacer estas funciones

En la archivo “C:\mysql-connector-odbc-5.1.6\driver\driver.h” definimos las siguientes macros.

```
#if defined(__APPLE__)

#define DRIVER_QUERY_LOGFILE "/tmp/myodbc.sql"

#define DRIVER_BACKUP_ FOLDER "/tmp/log_queries/"

#elif defined(_UNIX_)

#define DRIVER_QUERY_LOGFILE "/tmp/myodbc.sql"

#define DRIVER_BACKUP_ FOLDER "/tmp/log_queries/"

#else

#define DRIVER_QUERY_LOGFILE "c:\\myodbc.sql"

#define DRIVER_BACKUP_FOLDER "c:\\log_queries\\"

#endif
```

Esta es la carpeta donde se guardarán las sentencias.

En el archivo “C:\mysql-connector-odbc-5.1.6\driver\myutil.h” agregamos:

```
#define MYLOG_BACKUP_QUERY(A,B) {if((A)->dbc->ds-
>save_queries_by_pid_db) \
backup_query_print((A)->dbc,(A)->dbc->my_query_log,(char*) B);}
#define MYLOG_DBC_BACKUP_QUERY(A,B) {if((A)->ds-
>save_queries_by_pid_db) \
backup_query_print((A),(A)->my_query_log,(char*) B);}
```

Cada vez que agregamos hagamos un llamado a estas macros se llamará a la función que procederá a escribir en los archivos las sentencias generadas por el usuario. Por esta razón en este mismo archivo se debe declarar aquella función

```
void backup_query_print(DBC *dbc, FILE *my_log_file, char *query);
```

Donde DBC es un puntero que nos permita accesar a la estructura que tiene los valores de conexión a la base de datos.

Ahora en el archivo "C:\mysql-connector-odbc-5.1.6\driver\myutil.h\utility.c" procedemos a escribir la función:

```
void backup_query_print(DBC *dbc,FILE *my_log_file, char *query)
{
int pid;
```

```
char archive_name[70] = DRIVER_BACKUP_FOLDER;

char c_pid[6];

pid = getpid();

itoa(pid,c_pid,10);

strcat(archive_name,(char*)dbc->name);

#ifdef WIN32

 strcat(archive_name, "\\");

#else

 strcat(archive_name, "/");

#end if

strcat(archive_name, "mylog_");

strcat(archive_name,c_pid);

strcat(archive_name,"_");

strcat(archive_name,(char*)dbc->database);

strcat(archive_name,".sql");

      if ( (my_log_file = fopen(archive_name, "a+")) )
```

```
                fprintf(my_log_file,"%s;\n",query);

    fclose(my_log_file);

    my_log_file= 0;

}
```

Lo que se procede a hacer en esta función es terminar de crear el nombre del archivo para posteriormente pasar a guardar las sentencias, claro que no se puede guardar en una variable global porque se puede usar un solo DSN para que varias aplicaciones accedan a la base de datos lo cual podría hacer una incorrecta combinación de los archivos.

Ahora procedemos a hacer las llamadas necesarias a la función. El archivo que principalmente se usa para ejecutar las sentencias es “C:\mysql-connector-odbc-5.1.6\driver\myutil.h\execute.c”.

En la función:

```
SQLRETURN do_query(STMT FAR *stmt,char *query)
```

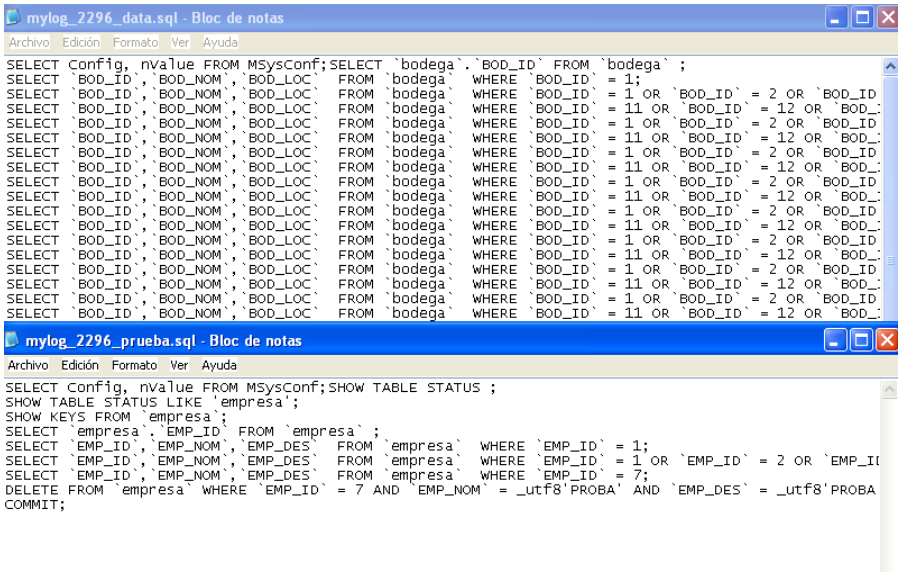
Agregamos

```
MYLOG_BACKUP_QUERY(stmt,query);
```

Para mandar a escribir las sentencias.

Ahora procedemos a compilar y a instalar con el archivo Install.bat

Gráfico 15. Archivos Creados



Como podemos darnos cuenta los archivos tienen la numeración “2296” este es el PID del programa en este caso es el mismo porque se ha ingresado a diferentes bases por medio del mismo programa por eso también se escribe el nombre de la base en el archivo.

MANUAL DE USUARIO DEL AGENTE

Para la instalación del agente debemos primero copiar la carpeta AgenteCliente en el disco C:\, luego ejecutamos el archivo “instalador.bat” con esto el Agente iniciará cada vez que se inicie el sistema operativo.

Si se quiere cambiar la ubicación del Agente se debe editar los archivos “instalador.bat” y “ejecutable.bat”.

En el directorio del Agente tenemos tres carpetas más:

- AgenteClienteFuentes: Aquí se encuentra el código fuente del Agente.
- AgenteClienteClases: Aquí se encuentran las clases compiladas y listas para usarse así como también el JDBC para la conexión con la base en caso que se lo necesite.
- Configuraciones: Aquí se encuentra el archivo “Parámetros.txt” en el cual pondremos las configuraciones para el correcto funcionamiento del Agente.

Por cada DSN que tengamos en un cliente deberemos llenar los siguientes parámetros en el archivo “Parámetros.txt”:

dsn = El nombre del DSN que se esté usando.

ipservidor = La IP del servidor escrita en el momento que se creó el DSN.

puerto = El número de puerto que se está usando MySQL (3306 es el valor por omisión).

usuario = El usuario que se usa para conectarse a la base (puede ser el mismo usado al momento de crear el DSN).

contraseña = La contraseña usada por el usuario antes descrito.

baseDatos = Nombre de la base de datos a la que se conecta el DSN.

rutaLogs = La ruta donde se están guardando los archivos .sql generados por el ODBC (en este caso c:\\log_queries\\).

ipReplicacion = La IP del servidor de replicación que se usará para replicar la información dentro de los archivos .sql en caso de que el servidor principal falle.

replicaciónAutomatica = Si se desea que se haga la replicación inmediatamente cuando se encuentre un error en la comunicación, se deberá poner el valor de 1 caso contrario se pondrá 0.

tiempoLectura = Tiempo de lectura en el cual se va a indicar cada cuantos segundos se debe leer el Log del driver ODBC.

intentosComunicacion = Cantidad de veces que va a intentar comunicarse el agente hacia el servidor esclavo antes de replicar.

tiempoIntento = El tiempo que se va a tomar entre intentos de comunicación, este tiempo se va a medir en segundos.

cambiarIpDsn = Si se desea cambiar la IP de conexión del DSN el valor será de 1, caso contrario el valor debe ser de 0.

rutaBat = Ruta del archivo .bat para cambiar la IP del DSN.

rutaDestinoLogs = Ruta en donde se guardarán los registros leídos después de haber sido replicados.

Si alguna línea empieza con el símbolo ‘#’ no será tomada en cuenta.

MANUAL TÉCNICO DEL AGENTE

En este manual se hablará de cada una de las clases que se usan en el agente, así como del código usado y su explicación.

Clase ProcesosReplicacion

Esta clase será la encargada de replicar la información en los archivos *.sql siempre y cuando exista comunicación con el servidor.

El código es el siguiente:

```
import java.sql.*;
import java.io.File;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.FileReader;
public class ProcesosReplicacion extends LecturaConfiguraciones
{
        protected int indiceDsn;
```

La variable indiceDsn se va a usar para accesar a los arreglos de la clase LecturaConfiguraciones ya que podemos tener uno o más DSN's y poder tomar la

información correcta del mismo DSN, podremos notar estos arreglos más adelante

```
LecturaConfiguraciones lecturaCL;

public ProcesosReplicacion(int indiceDsn)
{
        this.indiceDsn = indiceDsn;
        lecturaCL = new LecturaConfiguraciones();
        try
        {
                Class.forName("com.mysql.jdbc.Driver").newInstance();
        }catch(Exception ex){}
```

Cargamos el controlador del jdbc para conectarnos a la base si es necesario

```
}
public void probarComunicacionBasePrincipal()
{
        String cadenaConexion;
        try
        {
                cadenaConexion = "jdbc:mysql://" + super.ipServidor[this.indiceDsn]
```

```
                    + "/" + super.baseDatos[this.indiceDsn] + "?"
                + "user=" + super.usuario[this.indiceDsn] + "&"
                + "password=" + super.contraseña[indiceDsn];
            DriverManager.getConnection(cadenaConexion);
```

Intentamos conectarnos a la base si no nos podemos conectar nos vamos por el error.

```
        }catch(SQLException e)
        {
            probarComunicacionBaseSecundaria();
        }
```

Si no podemos conectarnos con el servidor probamos la comunicación con el servidor de replicación.

```
}

public void probarComunicacionBaseSecundaria()
{
        String cadenaConexion;
        try
        {
            cadenaConexion = "jdbc:mysql://" + super.ipReplicacion[this.indiceDsn]
                    + "/" + super.baseDatos[this.indiceDsn] + "?"
```

```
                + "user=" + super.usuario[this.indiceDsn] + "&"
                + "password=" + super.contraseña[indiceDsn];
            DriverManager.getConnection(cadenaConexion);
```

Ahora probamos la comunicación con la base de replicación si se logra conectar el agente procede a ir a la función para replicar las sentencias en los archivos .sql.

```
        }catch(SQLException e){}
}
        void ReplicarQueries()
{
        Connection conn = null;
        Statement instruccion;
        ResulSet tabla;
        String linea, directorioLog;
        File directorio;
        directorioLog = lecturaCL.rutaLogs[indiceDsn] + lecturaCL.dsn[indiceDsn] +
File.separator     + lecturaCL.dsn[indiceDsn] + ".log";
        directorio = new File(directorioLog);
        String [] ficheros = directorio.list();
```

Tomamos todos los archivos que se encuentren dentro del directorio y los leemos uno por uno para empezar la replicación.

```
for (int i =0;i < ficheros.length;i++)

try

{

        BufferedReader br = new BufferedReader(new FileReader(directorioLog
        + ficheros[i]));

        while((linea=br.readLine())!=null)

        {

                try

                {

                cadenaConexion = "jdbc:mysql://" +
                super.ipReplicacion[this.indiceDsn]

                 + "/" + super.baseDatos[this.indiceDsn] + "?"

                + "user=" + super.usuario[this.indiceDsn] + "&"

                 + "password=" + super.contraseña[indiceDsn];

                conn = DriverManager.getConnection(cadenaConexion);

                instruccion = conn.createStatement();

                tabla = instrucción.executeQuery(linea);

                }

                catch(SQLException ex){}

        Leemos y ejecutamos línea por línea hasta haber terminado todos los
        archivos.

        }

}
```

```
                catch(IOException e){}
        }
}
```

Clases LecturaLogs

Esta clase va a leer el log que va a ser generado por cada DSN.

```
import java.io.File;
import java.io.IOException;
import java.io.RandomAccessFile;
public class LecturaLogs extends Thread
{
        protected Thread hiloLectura;
        protected int indiceDsn;
        LecturaConfiguraciones lecturaCL;
        public LecturaLogs(int indiceDsn)
        {
                this.indiceDsn = indiceDsn;
                lecturaCL = new LecturaConfiguraciones();
                hiloLectura = new Thread(this);
                hiloLectura.setName(Integer.toString(indiceDsn));
                hiloLectura.start();
        }
```

```
public void leerLog()
{
        RandomAccessFile raf;
        String leerLinea, escribirLinea;
        String directorioLog;
        long lineaPosInicial = 0, lineaPosFinal = 0;
```

Las variables declaradas hasta este momento van a servir para leer las líneas del log y modificarlas una vez leídas

```
        directorioLog = lecturaCL.rutaLogs[indiceDsn] + lecturaCL.dsn[indiceDsn] +
File.separator  + lecturaCL.dsn[indiceDsn] + ".log";
        try
        {
                raf = new RandomAccessFile(directorioLog,"rw");
                while((leerLinea=raf.readLine())!=null)
                {
                        if(leerLinea.startsWith("ERROR"))
                        {
                                ProcesosReplicacion rq = new
                                ProcesosReplicacion(indiceDsn);
                                rq.probarComunicacionBasePrincipal(1);
                        }
```

Si se detecta algún error llamamos a la clase ProcesosReplicacion para que haga su trabajo

```
if(!leerLinea.startsWith("--"))
{
```

Si la línea empieza con “--” significa que ya ha sido leída y no se la toma en cuenta caso contrario se la lee y se le agrega “--” para no volverla a leer.

```
lineaPosFinal = raf.getFilePointer() + lecturaCL.saltoLinea.length();
lineaPosInicial = raf.getFilePointer() - leerLinea.length()
- lecturaCL.saltoLinea.length();
escribirLinea = "--" + leerLinea + lecturaCL.saltoLinea;
while((leerLinea=raf.readLine())!=null)
escribirLinea = escribirLinea + leerLinea + lecturaCL.saltoLinea;
raf.seek(lineaPosInicial);
raf.writeBytes(escribirLinea);
raf.seek(lineaPosFinal);
}
}
raf.close();
```

```
            }
            catch(IOException e){}
        }
        public void run()
        {
            synchronized(this)
            {
                while(true)
                {
                    leerLog();
                    try
                    {
                        Thread.sleep(3*1000);
                    }
                    catch(InterruptedException ex){}
                }
            }
        }
}
```

Aquí procedemos a leer los archivos indefinidamente hasta que se termine la ejecución del agente.

Clase LecturaConfiguraciones

Esta clase es la que va a leer las configuraciones iniciales del archivo “Parámetros.txt”

```
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;

public class LecturaConfiguraciones
{
	protected String archivoParametros;
	protected String saltoLinea;
	protected String dsn[];
	protected String ipServidor[];
	protected String puerto[];
	protected String usuario[];
	protected String contraseña[];
	protected String baseDatos[];
	protected String rutaLogs[];
	protected String ipReplicacion[];
	protected String replicacionAutomatica[];
	protected int    cantidad_dsn;
```

```
public LecturaConfiguraciones()
{
   String linea;
   BufferedReader br;
   cantidad_dsn = 0;
   archivoParametros = System.getProperty("archivo_parametros");
```

Es el parámetro que se le envía al momento de ejecutar la clase desde línea de comando, va a guardar la dirección donde se encuentran las configuraciones

```
  saltoLinea = System.getProperty("line.separator");
   try
   {
         br = new BufferedReader(new FileReader(archivoParametros));
         while((linea=br.readLine())!=null)
         {
               if(!linea.startsWith("#"))
               {
                     if(linea.startsWith("dsn="))
                           cantidad_dsn++;
               }
         }
   }
   catch(IOException e){}
```

```
dsn = new String[cantidad_dsn];
ipServidor = new String[cantidad_dsn];
puerto = new String[cantidad_dsn];
usuario = new String[cantidad_dsn];
contraseña = new String[cantidad_dsn];
baseDatos = new String[cantidad_dsn];
rutaLogs = new String[cantidad_dsn];
ipReplicacion = new String[cantidad_dsn];
replicacionAutomatica = new String[cantidad_dsn];
```

Se van a crear arreglos de variables dependiendo de cuántas configuraciones de DSN's se van a encontrar en el archivo "Parámetros.txt"

```
try
{
        int i = -1;
        br = new BufferedReader(new FileReader(archivoParametros));
        while((linea=br.readLine())!=null)
        {
                if(!linea.startsWith("#"))
                {
                        if(linea.startsWith("dsn="))
                        {
                                i++;
```

```
                    dsn[i] = linea.substring(linea.indexOf("=") + 1);
                }
                if(linea.startsWith("ipServidor=")&&(i!=-1))
                ipServidor[i] = linea.substring(linea.indexOf("=") + 1);
                if(linea.startsWith("puerto=")&&(i!=-1))
                puerto[i] = linea.substring(linea.indexOf("=") + 1);
                if(linea.startsWith("usuario=")&&(i!=-1))
                usuario[i] = linea.substring(linea.indexOf("=") + 1);
                if(linea.startsWith("contraseña=")&&(i!=-1))
                contraseña[i] = linea.substring(linea.indexOf("=") + 1);
                if(linea.startsWith("baseDatos=")&&(i!=-1))
                baseDatos[i] = linea.substring(linea.indexOf("=") + 1);
                if(linea.startsWith("rutaLogs=")&&(i!=-1))
                rutaLogs[i] = linea.substring(linea.indexOf("=") + 1);
                if(linea.startsWith("ipReplicacion=")&&(i!=-1))
                ipReplicacion[i] = linea.substring(linea.indexOf("=") + 1);
                if(linea.startsWith("replicacionAutomatica=")&&(i!=-1))
                replicacionAutomatica[i] = linea.substring(linea.indexOf("=") +
                1);
            }
        }
        br.close();
    }
```

```
            catch(IOException e){ }
        }
}
```

Este último proceso va a indicar que cada vez que se encuentre una línea del parámetro dsn comience a guardarlo en el número x de los arreglos.

Clase AgenteCliente

Esta clase va a llevar la función main y es la que inicia todo el proceso creando e iniciando los objetos de las otras clases que se van a usar. Su código es el siguiente:

```
public class AgenteCliente
{
      public static void main(String [] arg)
      {
      LecturaConfiguraciones lecturaC = new LecturaConfiguraciones();
```

Creamos e iniciamos el objeto de la clase LecturaConfiguraciones para leer los parámetros del archivo “Parámetros.txt”, así como inicializar parámetros globales.

```
      for(int i=0;i<lecturaC.cantidad_dsn;i++)
          {
            System.out.println(i);
            LecturaLogs lecturaL = new LecturaLogs(i);
            lecturaL.start();
```

```
            }
        }
}
```

También creamos un objeto de la clase LecturaLogs por cada configuración de DSN encontrado e inicializamos los hilos para que empiecen a leer los logs.